# Workflow Systems for LQCD

SciDAC LQCD Software meeting, Boston, Feb 2008

Fermilab, IIT, Vanderbilt

# What is a workflow?

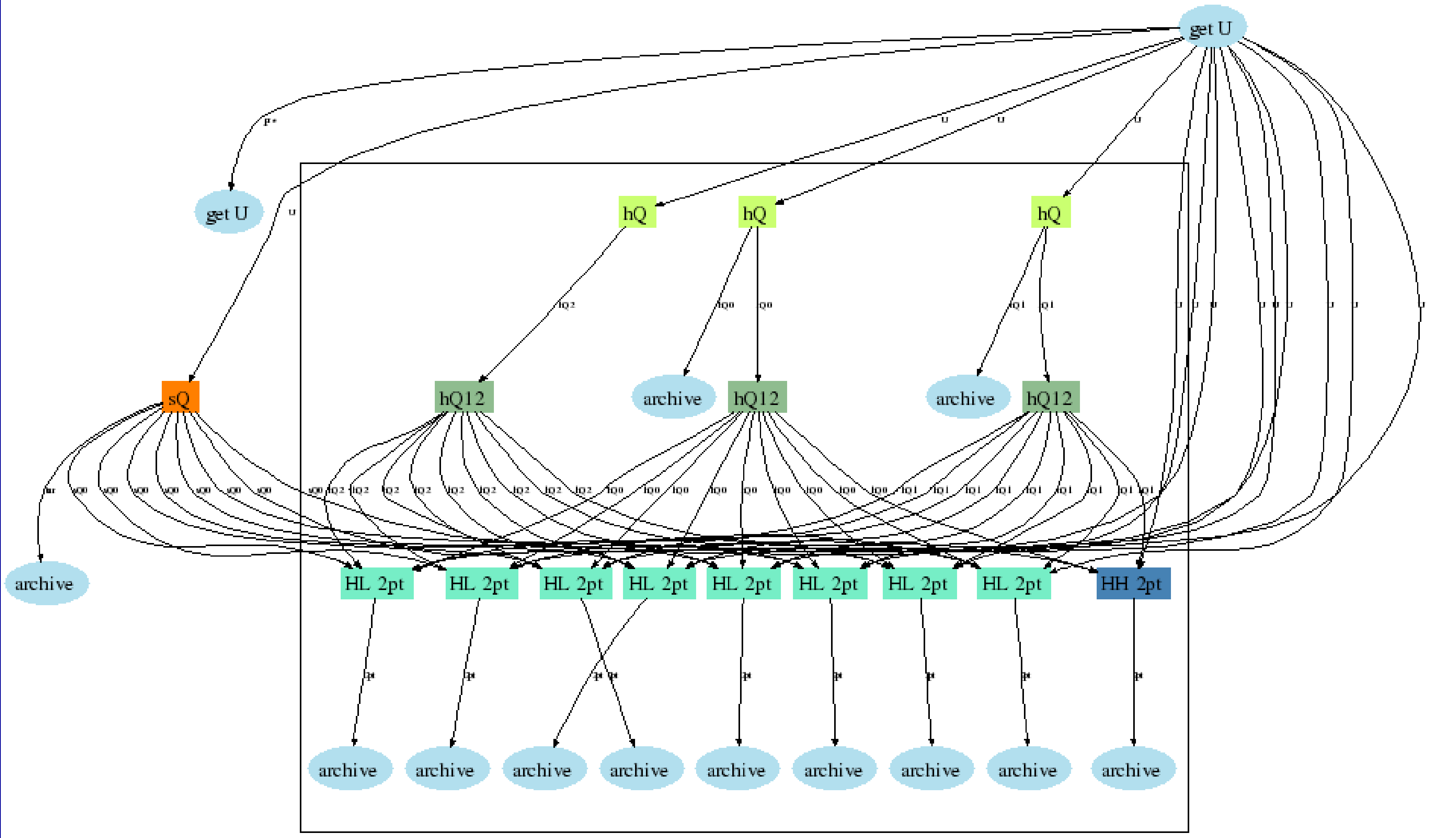
- A workflow is a reliably repeatable pattern of activity enabled by a systematic organization of resources, defined roles and mass, energy and information flows, into a work process that can be documented and learned. Workflows are always designed to achieve processing intents of some sort, such as physical transformation, service provision, or information processing.

- Hmm... mass? … energy? Must be physics related!

- A complex LQCD workflow: an LQCD analysis *campaign* to compute  hadronic 2- and 3-pt functions for each configuration of a gauge ensemble.

# LQCD 2-pt workflow

Foreach config in ensemble :



## Scientific workflow management systems

- Choices: Swift, Askalon, Kepler, Pegasus, Karajan, Triana, Taverna, *et al.*
- "*A Taxonomy of Workflow Management Systems for Grid Computing*"
  http://www.gridbus.org/papers/WorkflowTaxonomy-JoG.pdf
- Systems designed for Grid computing.
- Are systems easy to use? **NO**
  - Learning curve, added software complexity (and organization!).
- But I'm happy writing shell scripts, why change?
  - We envision compelling benefits in using a workflow system...

# Key workflow system requirements

- Users can exploit capacity computing while minimizing the burden of manually monitoring hundreds of batch jobs.
- Efficient use of LQCD compute facilities by monitoring and managing resources at the workflow level rather than only at the batch job level.
- Make a workflow specification easy to comprehend, reuse and extend.
- Eliminate the need for re-running any completed work upon resuming a stopped (by plan or by exception) analysis campaign.
- Campaign execution histories; use to plan future campaigns.
- Record provenance of scientific results.
- LQCD workflow management system requirements document http://whcdf03.fnal.gov/exp/attachments/WorkflowProject/FunctionalRequirements.doc
- Static dump of our wiki: http://whcdf03.fnal.gov/exp/WorkflowProject.html

# Highlights of the past year (or so)

- Wrote a functional requirements document

- On-going survey workflow systems:
  - Kepler tutorial at SC06, video meeting with Kepler team.
  - Face-to-face meetings with Swift team (ANL/U Chicago)
  - Video and face-to-face meeting with Askalon team (Innsbruck)
  - Phone meeting with Pegasus team (USC)
  - Presentation by developer from Condor and DAGman team

- Selected two promising systems for a detailed evaluation
  - Swift and Askalon
  - With help from developers, installed on Fermilab clusters
  - Coded a 2-pt LQCD workflow with help from developers
  - Proposed a simpler collection of workflow “unit tests”
  - Wrote a report detailing the evaluation

- A toy model workflow system in python
  - Database schema to persist state and data provenance

## From the Swift/Askalon evaluation executive summary

None of the currently available workflow systems can be quickly adapted for production use by LQCD, as several key requirements of the LQCD projects are not addressed by any of those systems. Additional development of specialized modules needs to be performed in-house to adopt these generic workflow systems for production use by the LQCD project.

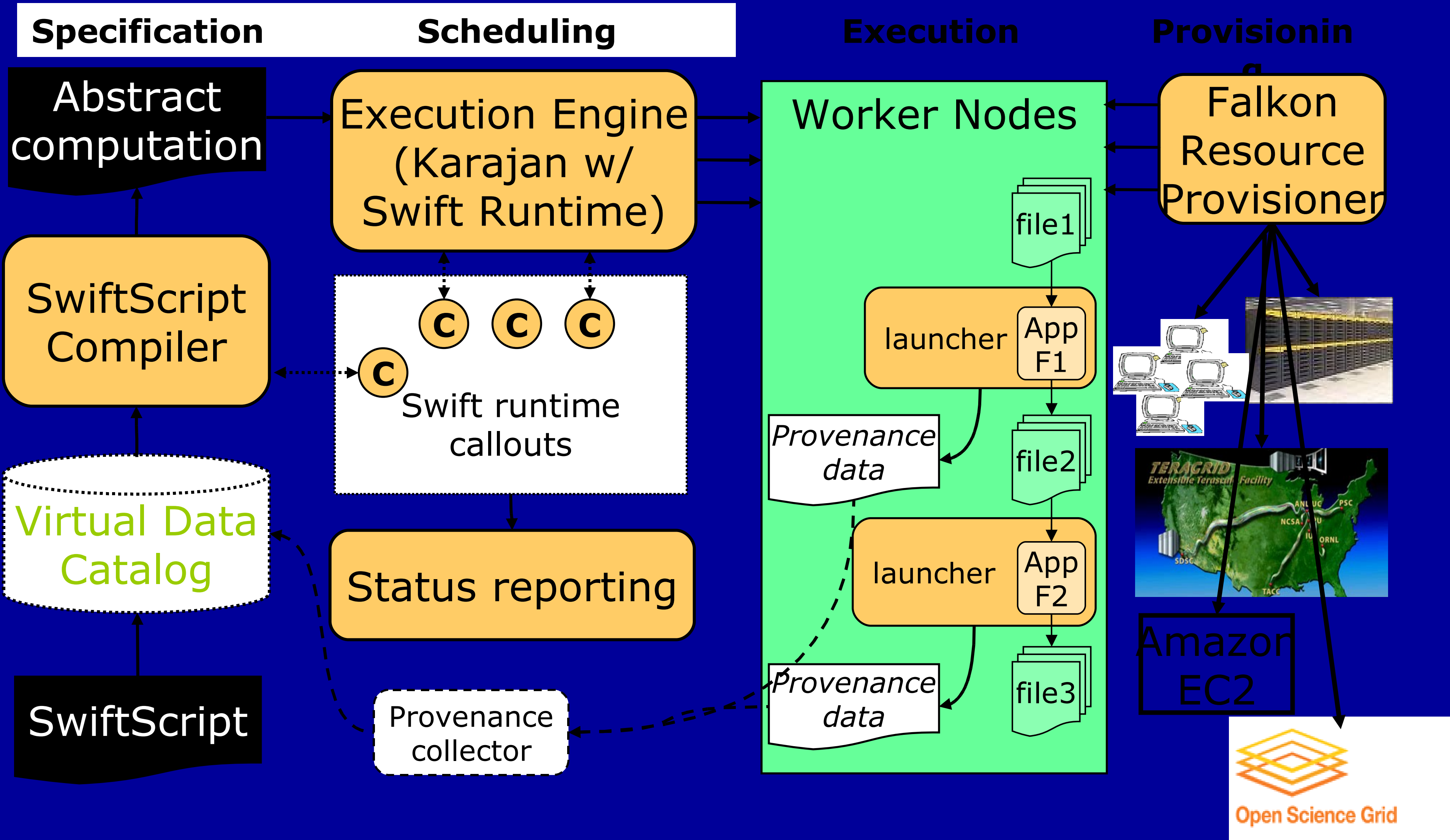
The advantage of using a currently available workflow system is that we could skip the initial and costly framework design and coding phase, thus spending out coordinated efforts on developing extensions as opposed to a completely custom solution. In making use of any existing workflow system there is an associated learning curve, but it is critical that we understand each workflow system's architecture so that we can be modify and/or effectively use those workflow system's APIs.

http://whcdf03.fnal.gov/exp/attachments/WorkflowProject/WorkflowEvaluation.doc

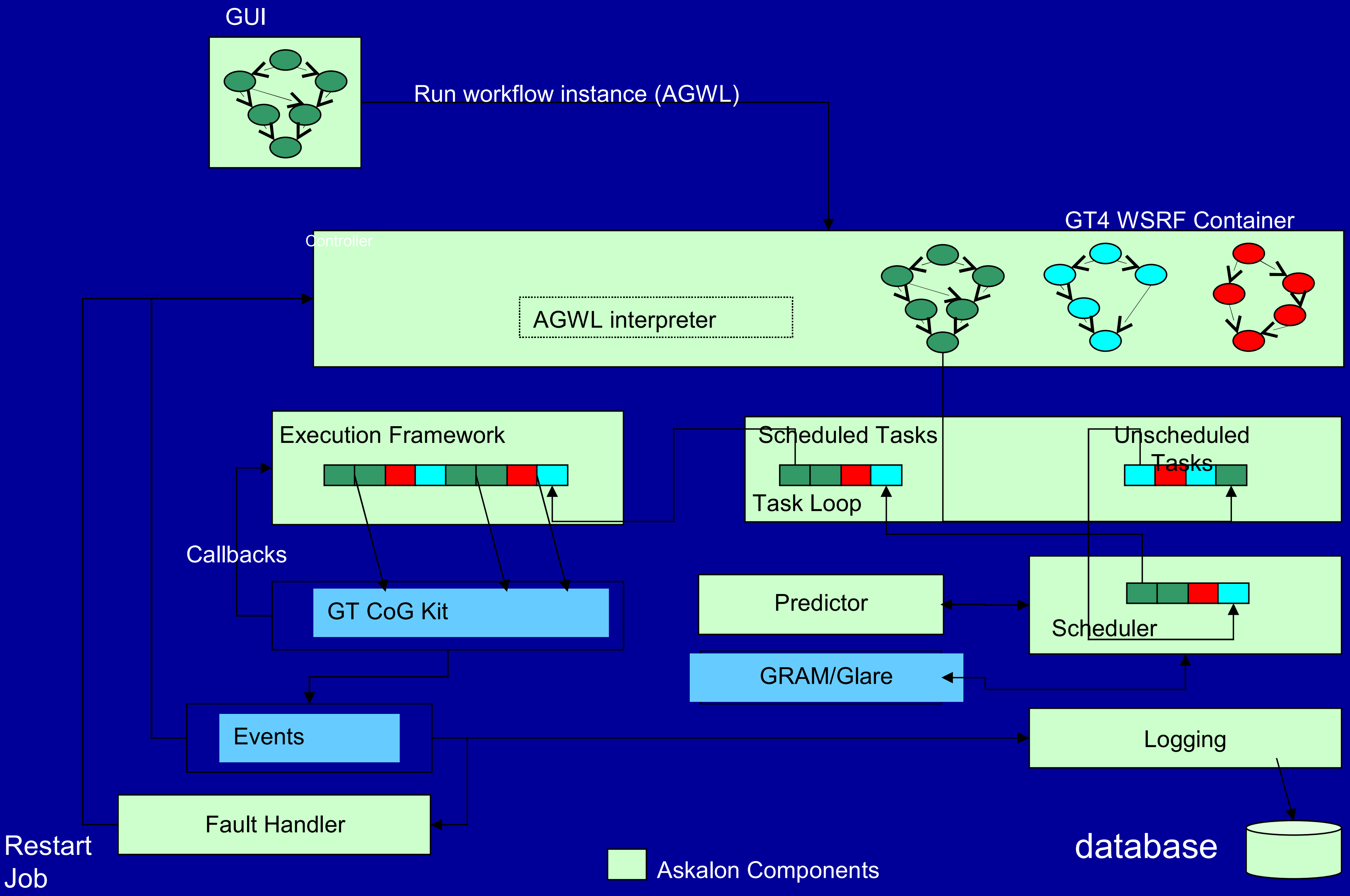
## Front- and back-end systems work

- Workflow composition and management
  - Management of workflow templates
  - Management of participants and their mapping to instances of runable code
  - Management of parameter sets (physics and system related)
- Scheduling and execution
  - Persistent workflow state
  - Fault tolerance (cluster reliability project)
  - Multiple workflows
  - Workflow and job (two-level) scheduling
- Workflow histories and data provenance
  - Management and user query facilities

# Swift Architecture



# Askalon architecture



🟩 Askalon Components

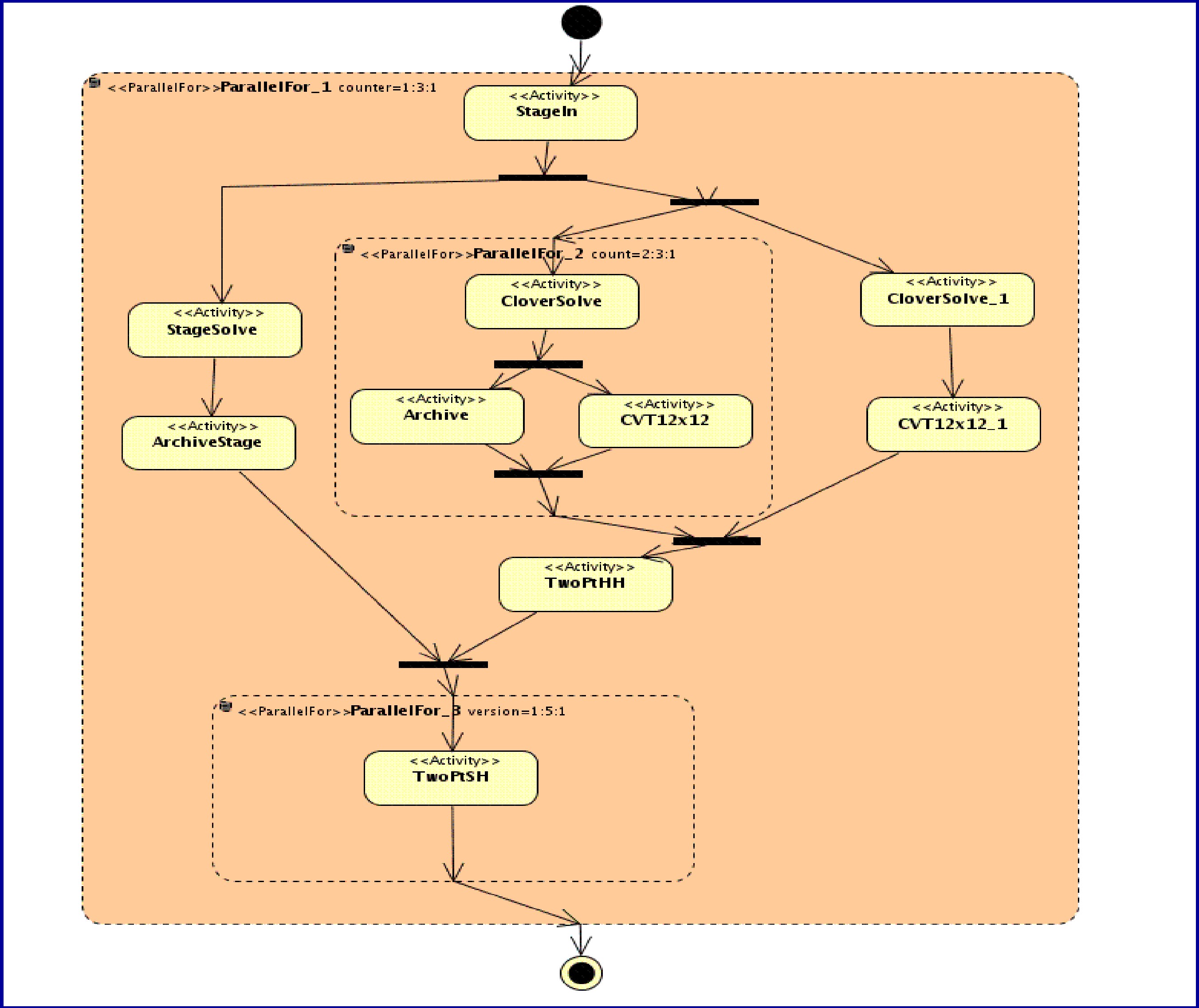
# Features overview

| **Feature** | **Swift** | **Askalon** |
|---|---|---|
| modeling language | swiftscript (karajan XML) | UML graphs (AGWL) |
| execution model | data flow | control flow |
| workflow scheduler | internal to swift / karajan | internal or ext. to GUI |
| users per session | single | single |
| workflows / session | one | one |
| QoS / fault tolerance | no / retry participants limited workflow restarts | perform. prediction / retry limited workflow restarts |
| execution history | log files | relational DB |
| data provenance | dot files | DB? |

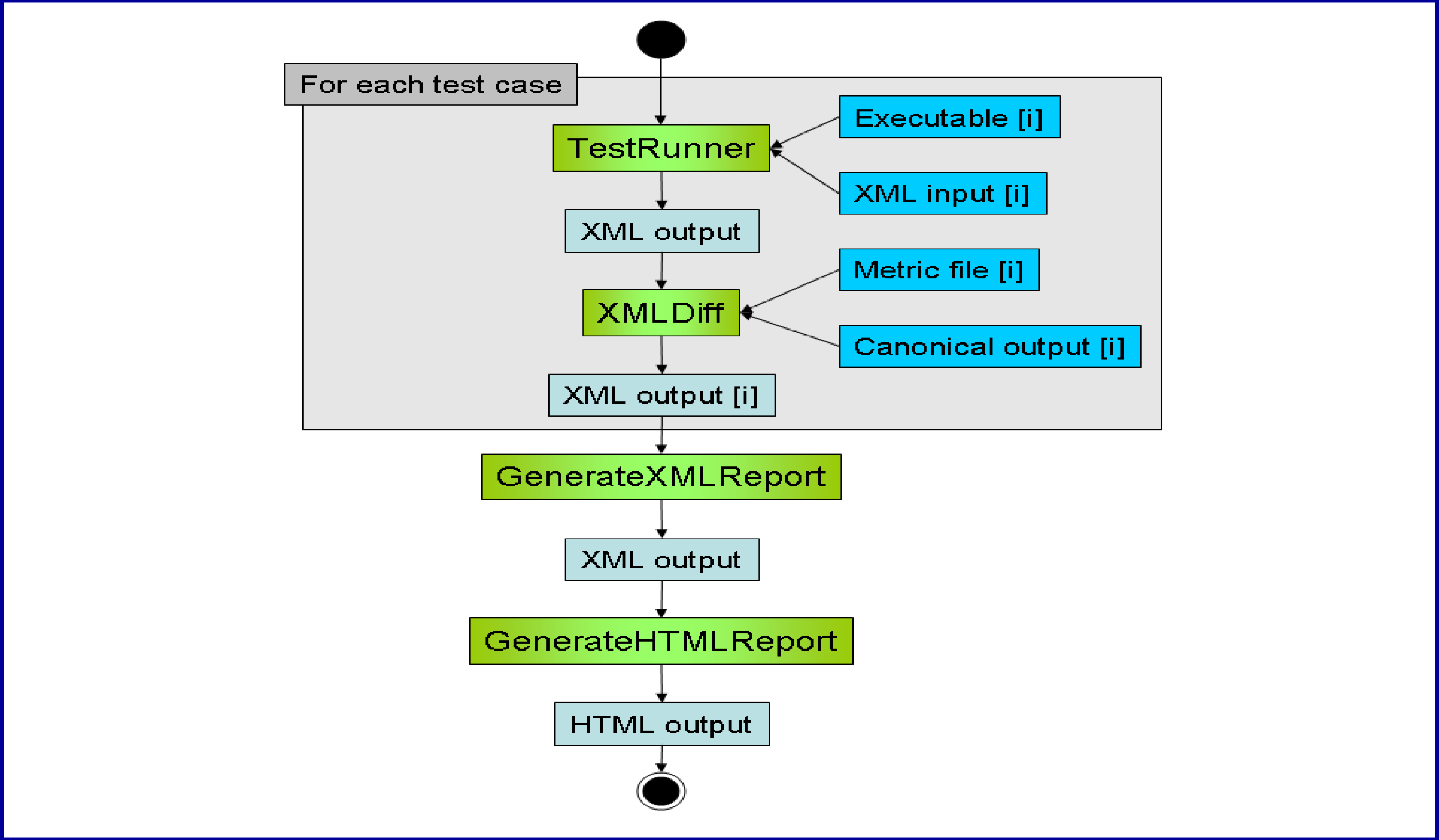
# HL 2-pt in swiftscript

```
String[] confList=["000102","000108"];
String kappaQ = "0.127";
String cSW = "1.75";
String mass = "0.005,0.007,0.01,0.02,0.03";
String[] fn = ["m0.005_000102 m0.007_000102 m0.01_000102 m0.02_000102 m0.03_000102",
               "m0.005_000108 m0.007_000108 m0.01_000108 m0.02_000108 m0.03_000108"];
String[] sources = ["local,0,0,0,0", "wavefunction,0,1S", "wavefunction,0,2S"];

foreach config,i in conflist {
          Template template <“foo">; # gauge template name
          Gauge gauge = stageIn(template, config);

          Stag stags[] <fixed_array_mapper; files=fn>;
          stags = stagSolve(gauge, mass, source);

          StagArchive stagTar <simple_mapper; suffix=".tar">;
          stagTar = archiveStag(mass, stags);

          Quark[] q;
          QuarkArchive[] cvtArch;
          for i in [0:2] {
                    Clover clover = cloverSolve(kappaQ, cSW, gauge, sources[i]);
                    q[i] = cvt12x12(clover);
                    cvtArch[i] = archive(q[i]);
          }

          Quark antiQ = q[0];
          file HH2ptcorr = twoPtHH(gauge, antiQ, q[0], q[1], q[2]);
          foreach stag in stags {
                    file SH2ptcorr = twoPtSH(gauge, stag, q[0], q[1], q[2]);
          }
}
```

# HL 2-pt in Askalon


<<ParallelFor>>ParallelFor_1 counter=1:3:1
<<Activity>> StageIn
<<Activity>> StageSolve
<<Activity>> ArchiveStage
<<ParallelFor>>ParallelFor_2 count=2:3:1
<<Activity>> CloverSolve
<<Activity>> Archive
<<Activity>> CVT12x12
<<Activity>> CloverSolve_1
<<Activity>> CVT12x12_1
<<Activity>> TwoPtHH
<<ParallelFor>>ParallelFor_3 version=1:5:1
<<Activity>> TwoPtSH

# Live demo: Chroma regression test



Code: http://home.fnal.gov/~piccoli/swift/SwiftRegres-x64-Jan30.tar.gz